

# Artificial Potential Biased Probabilistic Roadmap Method

Daniel Aarno, Danica Kragic and Henrik I Christensen  
Centre for Autonomous Systems  
Royal Institute of Technology  
Stockholm, Sweden  
Email:bishop@kth.se, {danik, hic}@nada.kth.se

**Abstract**—Probabilistic roadmap methods (PRMs) have been successfully used to solve difficult path planning problems but their efficiency is limited when the free space contains narrow passages through which the robot must pass.

This paper presents a new sampling scheme that aims to increase the probability of finding paths through narrow passages. Here, a *biased sampling scheme* is used to increase the distribution of nodes in narrow regions of the free space. A partial computation of the artificial potential field is used to bias the distribution of nodes.

## I. INTRODUCTION

Probabilistic roadmap methods (PRMs) have been successfully used to solve difficult path planning problems but their efficiency is limited when the free space contains narrow passages, [1]–[5]. The key idea of the basic PRM is to randomly distribute a set of nodes in the robot’s configuration space and then connect these nodes using a simple local planner (usually a straight line), to form a graph, known as a roadmap. This roadmap is supposed to capture the connectivity of the free space,  $C_{free}$ . If the roadmap is successful in capturing the connectivity of the free space, path planning may be reduced to a graph search.

The random sampling scheme used in basic PRMs does not work well when the configuration space contains narrow passages through which the robot must pass. This is because the random sampling scheme tries to distribute nodes with constant density in  $C_{free}$  and the volume spanned by the narrow regions of  $C_{free}$  makes up only a small fraction of the total volume spanned by  $C_{free}$ . Thus few nodes will end up in narrow regions, making it difficult for the planner to find a feasible path through the narrow passage. One way to solve this problem would be to bias the sampling in order to provide increased node density in narrow regions.

This paper presents a new planner, the artificial potential biased PRM (APBPRM). This planner uses a biased sampling scheme that can improve the performance of PRM planners in situations where the robot must pass through narrow passages. The key idea behind this new planner is to use an artificial potential field to increase the node density close to obstacle borders and especially in narrow regions. The artificial potential field is computed from a partial solution of Laplace’s equation (section IV-B) in the workspace  $W$  of the robot. Information from this potential field is then used to increase

the node density along  $C$ -obstacle surfaces, and especially in narrow and concave regions.

This paper is organized as follows. In Section II, a short review of the related work is given. In Section III, the properties of our method are described, and basic methodology is outlined in Section IV. The implementation is discussed in Section V. Experimental evaluation and results are presented in Section VI. Finally, we conclude with a short summary and suggestions for future work in Section VI-B.

## II. RELATED WORK

A number of schemes dealing with the narrow passage problem in PRMs have been proposed [1], [2], [5], [6]. They usually work by first distributing nodes uniformly throughout  $C$  and then, using information attained from this sampling of  $C$ , enhance the roadmap. This enhancement is done in different ways.

In [5], if the roadmap is disconnected in places where  $C_{free}$  exists, this place is considered to correspond to some narrow passage or a difficult region of  $C_{free}$ . Nodes in such regions are then expanded. Expanding a node  $\mathbf{q}$  corresponds to adding more nodes in its neighborhood. All nodes in the roadmap are given a positive weight  $w(\mathbf{q})$  which is a heuristic measurement of the “difficulty” of the neighborhood of  $\mathbf{q}$ . Thus,  $w(\mathbf{q})$  is larger whenever  $\mathbf{q}$  is considered to be in a difficult region of  $C_{free}$ . With  $w$  normalized ( $\sum_{\forall \mathbf{q}} w(\mathbf{q}) = 1$ ), nodes are repeatedly selected from the roadmap with probability  $P(\mathbf{q} \text{ is selected}) = w(\mathbf{q})$  and then expanded. Several ways to define the heuristic  $w(\mathbf{q})$  are given in [5]. One of these are similar to the method discussed in [4], using a  $w(\mathbf{q})$  that is inversely proportional to the number of neighbors. This method has the drawback that collision detection, roadmap construction and roadmap search have to be performed several times.

The planner in [1] uses the notion of dilated free space to increase the density of nodes near the boundary of  $C_{free}$ . This means that  $C_{free}$  is expanded, allowing the robot to “penetrate” a certain distance into obstacles. Nodes are then distributed in this dilated free space. Those nodes found to lie outside  $C_{free}$  are then “pushed” into  $C_{free}$  by a local resampling operation. This method would presumably fail given a task where very thin objects are present, making it impossible to expand  $C_{free}$ .

In [2], an obstacle-based PRM (OBPRM) planner is considered. This planner tries to add sample points close to or on  $C$ -space obstacle surfaces, similar to our APBPRM. The OBPRM described in [2] works in three steps. First, there is the node generation step, in which nodes are distributed in  $C$  in a way that increase the nodes density at  $C$ -obstacle surfaces. This is accomplished by finding configurations  $\mathbf{q}_i$  that intersect with  $C$ -obstacles (i.e.  $\mathbf{q}_i \notin C_{free}$ ). From these configurations, “rays” are shot out in random directions and the bounding surface of the  $C$ -obstacle is located by means of binary search. The second step is the roadmap connection where several more powerful local planners are used. First, the simple straight line planner is used to connect the nodes in  $C$ , and then, in regions found to be difficult, more advanced (and hence slower) planners are used. In the third step, the more powerful planners may also add new nodes to the roadmap, increasing the connectivity of the roadmap. This OBPRM actually deals with a quite general path planning problem with obstacles and APBPRM could easily be incorporated into this general planner in the node distribution step, possibly reducing the number of times the more advanced planners need to be invoked.

### III. PROPERTIES OF APBPRM

Because of the probabilistic completeness of PRMs (section IV-E), they can solve any problem given that a sufficiently large number of enhancements are made to the roadmap. This fact implies that a biased sampling scheme might not be necessary. However, this is not entirely true. One obvious reason to prefer a biased sampling scheme is that graph search time is reduced, since for every enhancement step the roadmap has to be rebuilt and searched again. Also, the enhancement steps often tend to oversample the “open” regions of  $C$ , creating a roadmap with more nodes than is actually required to solve the problem (increasing search time). Another issue arises when there are two (or more) ways to reach the target and one way is shorter than the other but contains a narrow passage (Figure 1) .

A classical PRM planner could probably find the “long way around” (dashed line in Figure 1) fairly easy, given a suitable number of initial nodes in the roadmap. If the planner finds a solution to the path planning problem, it will not perform an enhancement step. While this behavior might be acceptable in some cases, it will sometimes be more important to find the shorter path. There is, of course, a trade-off here. A robot can travel with a higher speed if it decided to take the “long way around” because it would have to be less concerned with bumping in to the walls, thus the time required is not directly related to the length of the path. If such aspects are taken into consideration, the function that measures the “goodness” of a path in  $C$ -space might have to be changed to penalize paths that are too close to obstacle borders. The graph searching algorithm should then optimize on this “goodness” function rather than the  $C$ -space distance.

From the planners perspective, it is better to find both paths and then choose the best, according to some metric. The

APBPRM would have a much higher probability of finding the narrow passage path (solid line in Figure 1), given the same number of initial nodes as the classical PRM planner.

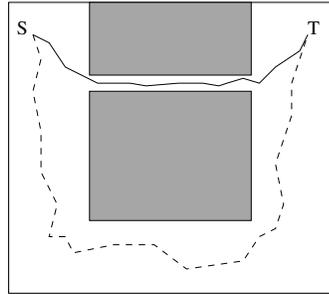


Fig. 1. A point shaped robot planning a path from S to T in an environment containing a narrow passage. APBPRM yields the solid line path and standard PRM yields the dashed line path.

#### A. Computational Benefits

Using the potential function to bias sampling in a PRM planner could also provide some computational benefits. First of all, since the APBPRM planner is less likely to perform the enhancement step, roadmap connection and search time could be reduced. Since node distribution time usually requires a few percent of the total computation and the rest is used for roadmap connection/search, it would be preferable to minimize the number of connections and searches made. Although APBPRM adds somewhat to the time required for distributing nodes, it is presumably better than performing one or more extra connect/search steps.

Since APBPRM uses a partial solution of Laplace’s equation to bias the search, one could easily imagine a scheme where a more accurate solution of Laplace’s equation is computed (more steps in the FDM (Finite Difference Method, see section IV-B) solution). This solution could then be used for gross motion planning or to guide the search algorithm, i.e. search “down-hill” first. Using this scheme would minimize the time “wasted” when computing the partial solution to Laplace’s equation. However these effects are not considered in this paper.

### IV. METHODOLOGY

This section provides a formal description of the path planning problem, introducing the reader to the notation used in this paper. This is followed by the theoretical foundation of the APBPRM planner.

#### A. The Path Planning Problem

Let  $A$  denote an arbitrary robot, (an agent) consisting of one or more rigid bodies with  $N$  degrees of freedom (dof). The path planning problem solves the problem of connecting, by a continuous path, a point  $\mathbf{q}_{start}^c$  to any point  $\mathbf{q}_{goal}^c \in C_{free}$  that satisfies the condition:  $\mathbf{q}_{goal}^c \rightarrow \mathbf{q}_{goal}^w$  under the constraints of the forward kinematics of  $A$ . Here  $\mathbf{q}^c$  denotes a point in the robot’s configuration space and  $\mathbf{q}^w$  denotes a

configuration (position and orientation of a specific part of the robot) in the robot's workspace.  $C_{free}$  is the subset of the robot's configuration space available to the robot, i.e. not occupied by obstacles.

### B. Theoretical considerations

Another type of planners, known as potential field planners [7]–[9] use gradient decent on a potential function  $\phi(\mathbf{q})$  defined over  $C$  to solve the path planning problem. In general, Laplace's equation:

$$\nabla^2 \phi(\mathbf{x}) = \sum_{i=1}^N \frac{\partial^2 \phi}{\partial x_i^2} = 0 \quad (1)$$

can be used to describe the potential of a particle in free space acted on only by gravitational forces, [10] which is the property artificial potential methods try to mimic. Because of this property, Laplace's equation is often referred to as the potential equation.

Solving (1) in  $C_{con}$  with the potential  $\phi$  held fixed at 1 (Dirichlet boundary condition [10], [11]) on all points  $\mathbf{q}^c \notin C_{free}$  and at  $-1$  on all points  $\mathbf{q}^c \rightarrow \mathbf{q}_{goal}^w$ , will result in an artificial potential field  $\phi_c$  in  $C_{con}$ . Performing gradient descent on  $\phi_c$  will result in a path from the starting point  $\mathbf{q}_{start}^c$  to a minima  $\mathbf{q}_{min}^c$ . If  $\mathbf{q}_{min}^c \rightarrow \mathbf{q}_{goal}^w$  (a solution has been found) the planner is done. However, if  $\mathbf{q}_{min}^c \nrightarrow \mathbf{q}_{goal}^w$  (a solution has not been found),  $\mathbf{q}_{min}^c$  is a local minima that must be escaped from. This escape can be achieved using of a random walk or a local search. After the escape from a local minima, a new gradient descent may be performed and a new minima is located. This process can then be repeated until a solution has been found.

One drawback of the potential field approach is the creation of local minima that does not correspond to a goal configuration. If the world in which the robot operates is complicated, containing many objects or objects of complex shape, the robot may get stuck moving from one local minima to another not reaching the goal in the required time. Another problem is that, as the dimensionality of  $C$  increases, the time required to compute  $\phi$  grows rapidly. The problem with high dimensional  $C$  may be circumvented by computing  $\phi$  in  $W$ . Since  $1 \leq \dim(W) \leq 3$ ,  $\phi$  may always be computed in  $W$  in a relatively short time (compared to  $C$  that may have 10s of dimensions). Once the potential  $\phi^w(\mathbf{x})$  is known in  $W$ , the potential  $\phi^c(\mathbf{q})$  for a specific location in  $C$  may be computed by summing the potential for all points in the region  $\Omega \subset W$  occupied by  $A$ , [9]. This relation can be written as

$$\phi^c(\mathbf{q}) = \sum_{\forall \mathbf{x} \in \Omega} \phi^w(\mathbf{x}) \quad (2)$$

Equation (2) makes it possible to use potential fields for higher dimensions of  $C$  than it would otherwise be feasible. However, in order to perform gradient descent not only the potential of the current configuration must be known but also the potential of all neighboring configurations. The time

required to compute all those potentials will eventually grow to unacceptable values but it is still much better then computing  $\phi$  in  $C$  explicitly.

The potential function can be computed numerically using standard finite difference methods (FDM) [7]. Since (1) can be used to describe voltages in a resistive grid, a resistive grid can be used to obtain an analog solution to  $\phi$  in a matter of microseconds, [7], [11].

### C. The Narrow Passage Problem with Artificial Potential Fields

Artificial potential field planners performs well in relatively uncluttered workspaces. However, if the robot has to move through a narrow passage artificial potential field planners, just as PRM planners, experience problems. This is because the potential in a narrow passage will be high. If a local minima exists near the entrance of a narrow passage it is unlikely that the planner will be able to escape its minima through the high potential ridge in the narrow passage.

### D. APBPRM

When solving (1) numerically in the region  $\Psi$ , iterative methods such as Jacobi iteration, Gauss-Seidel iteration, Crank-Nicolsons method or Successive Over-Relaxation (SOR) can be used [7], [11]. For (1), these methods essentially replaces each grid point's value with a weighted average of its neighbors. This is then repeated iteratively until a stable solution is found.

Instead of computing the solution of Laplace's equation, APBPRM uses the idea that, while solving for the potential  $\phi$ , iterative methods will in general cause the potential to rise more rapidly in narrow regions. An intuitive way to visualize this is that a grid point that has a Manhattan distance of  $n$  to the closest boundary point will remain at (the initial) zero potential for the first  $n$  steps of the iterative computation. Grid points close to the boundary of  $\Psi$  can, on the other hand, be updated many times during the first  $n$  iterations and thus rise to a high potential. This is especially true for grid points surrounded by boundary points, such as grid points in narrow or concave regions of  $\Psi$ .

APBPRM computes  $\phi_N$ , the first  $N$  steps of a solution to  $\phi$  using FDM and uses this to bias the distribution of nodes in the roadmap. The node distribution scheme used by APBPRM works as follows:

First a set of nodes,  $\mathbf{Q}_{rnd}$ , is distributed at random uniformly throughout  $C$ , keeping only those nodes  $\mathbf{q} \in C_{free}$  until  $\mathbf{Q}_{rnd}$  contains  $M$  nodes. Then more nodes are randomly distributed in the same way but keeping only those nodes  $\mathbf{q} \in C_{free}$  with a probability  $P$  given by (3), where  $k_\phi$  and  $k_r$  are arbitrary real constants, until the set of nodes  $\mathbf{Q}_{apb}$  contains  $K$  nodes.

$$P(\mathbf{q} \text{ is kept}) = \begin{cases} 0 & \text{if } w(\mathbf{q}) < 0 \\ w(\mathbf{q}) & \text{if } 0 \leq w(\mathbf{q}) \leq 1 \\ 1 & \text{if } w(\mathbf{q}) > 1 \end{cases} \quad (3)$$

where  $w(\mathbf{q}) = k_\phi \phi_N(\mathbf{q}) + k_r$

Using the probability given by (3), *all* nodes are kept with at least the probability  $k_r$  (unless  $k_r < 0$  which might be interesting to investigate in some high dimensional cases), and the probability of keeping a node is increased proportionally to  $\phi_N$ . The set of nodes in the roadmap is finally constructed by combining the two sets of nodes to a new set  $\mathbf{Q} = \mathbf{Q}_{apb} \cup \mathbf{Q}_{rnd}$  that form the nodes of the roadmap. This will result in denser sampling of the  $C$ -space close to obstacle boundaries and especially in narrow and concave regions. The idea that a denser distribution of nodes along  $C$ -space surfaces helps to guide the robot through narrow passages is also supported by [2], [6].

### E. Probabilistic Completeness

In this section the probabilistic completeness of APBPRM is discussed. A path planner is called probabilistically complete if the probability of solving any given problem approaches 1 as the planning time approaches  $\infty$ , provided that a solution exists. A proof of the probabilistic completeness of PRM planners is given in [4].

Because APBPRM is a simple extension to normal PRM it has the same probabilistic completeness as other PRM planners. This means that APBPRM is able to solve any given problem for any given robot for which a solution exists, given sufficient running-time and that the robot is locally controllable, [12]. The property of local controllability is further discussed in [12] and essentially means that a robot  $A$  always can move in a neighborhood of  $\mathbf{q}$  for any given  $\mathbf{q} \in C_{free}$ .

## V. IMPLEMENTATION

To test the theoretical foundation of APBPRM, a sample PRM planner with support for artificial potential biased sampling was implemented. Pseudo code for the APBPRM planner is outlined in algorithm 1. The planner used in the experiments used the Lazy evaluation scheme presented in [4].

---

### Algorithm 1 A single path planning query.

---

```

world ← Load world from file
if( $\mathbf{q}_{start}$  or  $\mathbf{q}_{goal}$  is not valid)
  return FAILURE
Compute potential for world
nodes ← Distribute nodes according to policy
Add nodes to roadmap
Build graph in roadmap
while( $\mathbf{q}_{start}$  and  $\mathbf{q}_{goal}$  are connected)
  path ← Shortest path from  $\mathbf{q}_{start}$  to  $\mathbf{q}_{goal}$  in roadmap
  if(path is collision free)
    return path
  remove illegal edge and/or node in path from roadmap
end while
return FAILURE

```

---

The world is modeled as a uniform and variable resolution grid with the world coordinates normalized, i.e.  $x, y, z \in [0, 1]$ . A World object begins by loading a bitmap image representation of the world where the obstacles are marked

by a 1 and the free space is marked by a 0. Once the world representation is loaded the World object computes and stores  $\phi_N$ . The World class provide access to the partial potential for points in  $W$  (truncated to the nearest grid point) and a function that tests if a point in  $W$  lies in  $W_{free}$ .

A RoadMap object is provided with a list of nodes and a start and goal configuration. It begins by building the roadmap graph. All nodes, including the start and goal nodes, are inserted in an array and are provided with a unique hash key for efficient reference. In addition all nodes are provided with pointers to their adjacent nodes.

The complexity of building the graph is  $\mathcal{O}(n \log(n))$ , where  $n$  is the number of nodes in the roadmap. However, building the roadmap is a parallel process and can thus take advantage of multi processor machines. Once the graph is built, the RoadMap object can be queried for a solution to the path planning problem. The graph is now search for the shortest possible solution path using Dijkstra's algorithm [13]. Dijkstra's algorithm has  $\mathcal{O}((e + n) \log(n))$  complexity, where  $e$  is the number of edges and  $n$  is the number of nodes in the roadmap. Better algorithms that use a heuristic to guide the search, such as A\* search, exists [4], [14] but were not used because the behavior of a complete algorithm is easier to understand and analyze.

Once a path is found, it is checked for validity. The collision checking from [4] is used for high efficiency. If the path is valid the planner is done, if not the edges and nodes found to be illegal are removed and the graph is searched again. This is repeated until either a solution path is found or the goal and start configurations get disconnected. If the goal and start configurations get disconnected the planner reports failure. No enhancement step is implemented at this stage.

## VI. EXPERIMENTAL EVALUATION

To measure and compare the performance of APBPRM vs PRM, four path planning tasks with a point shaped robot in two dimensions were performed. Figure 2 shows the different workspaces of the planning tasks. Each task was performed 100 times with both biased and random sampling and the probability of reaching the goal without requiring an enhancement step was calculated. In addition, the average number of paths that were tested for collision was calculated. The planning task was performed using a different number of nodes in the roadmap (100, 250, 500, 750). The results of these simulations can be seen in Table I.

Results are averaged over 100 trials. The start and goal configurations were the same for every trial (although different for each world). In all simulations nodes were kept with a probability given by (3) with  $\phi_{100}$  and  $k_r = 0.1$ . When using the biased sampling scheme no nodes were distributed at random (i.e.  $\mathbf{Q}_{rnd}$  is empty) but rather the connectivity of open regions was captured using  $k_r = 0.1$ . The world was modeled as a  $180 \times 180$  grid. The number of neighbors were limited to a maximum of 100.

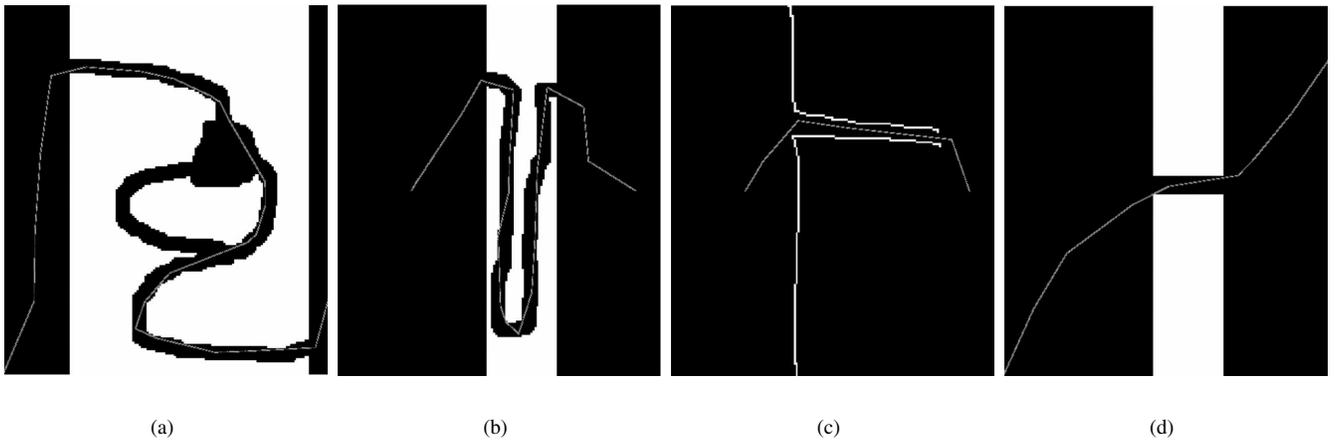


Fig. 2. The different worlds used to evaluate the performance of APBPRM, each world shows an example path.

TABLE I

SIMULATION STATISTICS FOR THE DIFFERENT WORLDS IN FIGURE 2.

N	S	A		B		C		D	
		GR	PT	GR	PT	GR	PT	GR	PT
100	RND	0	20	0	36	0	82	41	5
	APB	10	86	1	362	10	262	69	21
250	RND	43	287	2	277	45	361	81	18
	APB	84	865	68	2643	84	866	97	146
500	RND	92	1357	22	1305	90	789	97	59
	APB	99	3352	100	7505	99	1739	100	440
750	RND	100	2647	61	2953	98	1069	100	115
	APB	100	5887	100	11684	100	2381	100	597

GR: Chance of reaching the goal without requiring an enhancement step (%).  
 N: Nodes in the roadmap.  
 PT: Number of paths tested.  
 S: Sampling strategy. RND = uniform random sampling, APB = APB sampling.

### A. Experimental results for a 5 dof Planar Link Robot

This section shows some of the preliminary results of APBPRM when applied to a 5 dof planar arm. Because the planner uses a complete algorithm, it is not possible to use the amount of nodes needed (Because the planning time would be too long) to provide sufficient resolution in  $C$ -space, thus giving poor results. Also no investigation of the effect of the parameters  $k_\phi$  and  $k_r$  in (3) has been performed.

It is clear from the results that APBPRM can be used to increase the probability of finding paths through narrow passages for a point shaped robot. The success of APBPRM indicates that denser sampling around  $C$ -space obstacles surfaces aid the planner in narrow and cluttered regions.

At this stage, no effort to incorporate the gradient of the potential function into the roadmap search has been done. However, this is an interesting idea that will be further investigated. The preliminary results for a 5 dof planar arm indicate that the success of APBPRM decreases some with increased

TABLE II

PLANNING RESULTS FOR THE 5 DOF PLANAR ARM IN THE TWO WORLDS FROM FIGURE 3. AVERAGE RESULTS OVER 100 TRIALS.

Sampling	A		B	
	GR	PT	GR	PT
APB	78	14.9	30	18.0
RND	69	5.0	23	3.3

GR: Chance of reaching the goal without requiring an enhancement step (%).  
 PT: Number of paths tested.

dimensionality of the  $C$ -space. This is not necessarily true and will be investigated as part of future work. For instance the relatively poor performance reported in section VI-A could be due to a bad selection of parameters in (3). If  $k_r$  is too high the sampling of the  $C$ -space will be too gross, i.e. nodes are kept too easy. Combined with a uniform random sampling it might even be interesting to evaluate the performance with  $k_r < 0$ .

It is often easy to find situations where heuristic methods fail, this is also true for APBPRM. What APBPRM provides is another way of handling the narrow passage problem, increasing the set of “tools” available for people who construct motion planners to be used in real world applications.

Some of the drawbacks of the proposed approach is that since the potential function is usually quite steep near obstacles the planner will tend to “crawl” near the edges of obstacles. While this is not an issue during planning for a massless agent in an completely known environment, it is when planning motions for real physical robots in an approximation of the real world. In the real world, the robot requires some minimum clearance to the obstacles. Also, the robot has to move more slowly when close to obstacle boundaries to avoid the risk of collision. APBPRM may generate paths where the robot crawls along the edges of obstacles. The computation of the solution to Laplace’s equation in  $\mathbb{R}^3$  is quite expensive,

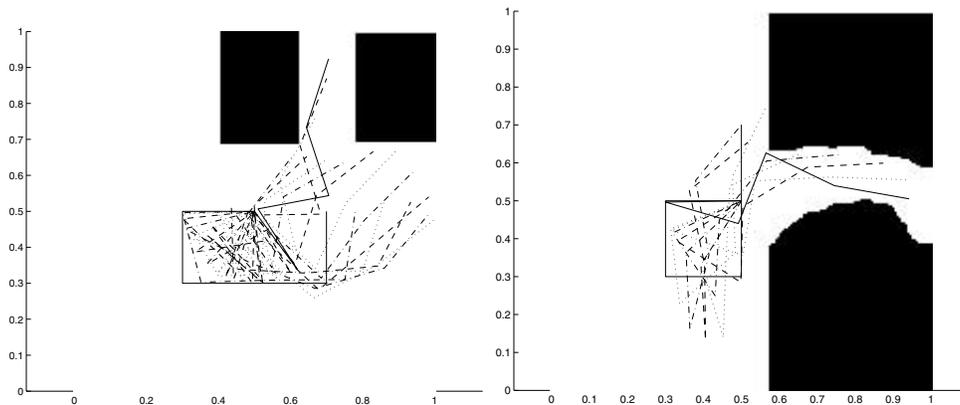


Fig. 3. Two tests with a 5 dof planar arm. The start and goal locations are shown (solid lines) as well as some intermediate positions.

limiting the usefulness of APBPRM in environments with moving obstacles. However methods for dealing with such cases exists and can be incorporated to the APBPRM planner [15], [16].

### B. Future Work

In this paper, we have presented a new sampling scheme used to increase the probability of finding paths through narrow passages. Here, a biased sampling scheme is used to increase the distribution of nodes in narrow regions of the free space. A partial computation of the artificial potential field is used to bias the distribution of nodes.

Some of the ideas presented in section IV-B have not been implemented or tested. One of the interesting future developments would be to investigate the performance gained when adding biased sampling to existing PRM planners and using their more efficient search algorithms. Also, the possibility to use the potential function as a heuristic in search algorithms such as A\*-search will require further investigation. The effect of the parameters  $k_\phi$  and  $k_r$  in (3) needs to be evaluated. Finally we will investigate modifications to (3). For instance using  $w(\mathbf{q}) = k_\phi \sqrt{\phi_N(\mathbf{q})} + k_r$  might lessen the tendency to crawl along edges.

### REFERENCES

- [1] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Robotics: The algorithmic perspective* (L. E. K. P. K. Agrawal and M. Mason, eds.), (Natick, MA), pp. 141–153, A.K. Peters, 1998.
- [2] Overmars and Svestka, "A probabilistic learning approach to motion planning," in *Algorithmic Foundations of Robotics, The 1994 Workshop on the Algorithmic Foundations of Robotics*, A. K. Peters (Goldberg, Halperin, Latombe, and Wilson, eds.), 1995.
- [3] J. Barraquand, L. Kavraki, J. Latombe, T. Li, R. Motwani, and P. Raghavan, "A random sampling scheme for path planning," in *Robotics Research* (G. Giralt and G. Hirzinger, eds.), pp. 249–264, Springer, 1996.
- [4] R. Bohlin, *Robot Path Planning*. PhD thesis, Chalmers university of technology, Göteborg university, 2002.
- [5] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," Tech. Rep. CS-TR-94-1519, 1994.
- [6] D. Hsu, T. Jiang, J. Reif, and S. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," *IEEE International Conference on Robotics and Automation*, 2003.
- [7] C. I. Connolly and R. A. Grupen, "Applications of harmonic functions to robotics," Tech. Rep. UM-CS-1992-012, Computer and Information Science Department, 1992.
- [8] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE on Robotics and Automation*, vol. 8, pp. 501–518, October 1992.
- [9] Y. K. Hwang and N. Ahuja, "Gross motion planning - a survey," *ACM Computing Survey*, vol. 24, pp. 219 – 291, September 1992.
- [10] W. E. Boyce and R. C. DiPrima, *Elementary Differential Equations and Boundary Value Problems*. John Wiley & Sons, Inc, sixth ed., 1997.
- [11] T. Erikson, H. Christiansson, E. Lindahl, J. Linde, L. Sandberg, and M. Wallin, *Fysikens Matematiska Metoder*. Teoretisk Fysik, KTH, third ed., 2001.
- [12] P. Svestka, "On probabilistic completeness and expected complexity of probabilistic path planning," tech. rep., Utrecht University: Information and Computing Sciences., 1996.
- [13] A. M. Tenenbaum, Y. Langsam, and M. J. Augenstein, *Data Structures Using C*. Prentice-Hall, Inc, 1990.
- [14] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc, 1995.
- [15] R. Kindel, D. Hsu, J.-C. Latombe, and S. Rock, "Kinodynamic motion planning amidst moving obstacles," in *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 537 – 543, 2000.
- [16] J. H. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," in *IEEE Symposium on Foundations of Computer Science*, pp. 144–154, 1985.